The Impact of Optimization Algorithms on The Performance of Face Recognition Neural Networks

Mahmoud ALI and Dinesh KUMAR*

Computer Engineering Department, Marwadi University, Rajkot, India

*Corresponding Author: Mahmoud ALI (Email: mahmoud.ali106127@marwadiuniversity.ac.in)
(Received: 6-Feb-2022; accepted: 14-Jun-2022; published: 31-Dec-2022)
DOI: http://dx.doi.org/10.55579/jaec.202264.370

Abstract. *Face recognition has aroused great interest in a range of industries due to its practical applications nowadays. It is a biometric method that is used to identify and certify people with unique biological traits in a reliable and timely manner. Although iris and fingerprint recognition technologies are more accurate, face recognition technology is the most common and frequently utilized since it is simple to deploy and execute and does not require any physical input from the user. This study compares Neural Networks using (SGD, Adam, or L-BFGS-B) optimizers, with different activation functions (Sigmoid, Tanh, or ReLU), and deep learning feature extraction methodologies including Squeeze Net, VGG19, or Inception model. The inception model outperforms the Squeeze Net and VGG19 in terms of accuracy. Based on the findings of the inception model, we achieved 93.6% of accuracy in a neural network with four layers and forty neurons by utilizing the SGD optimizer with the ReLU activation function. We also noticed that using the ReLU activation function with any of the three optimizers achieved the best results based on findings of the inception model, as it achieved 93.6%, 89.1%, and 94% of accuracy for each of the optimization algorithms SGD, Adam, and BFGS, respectively.*

Keywords

Activation Functions, Embeddings, Face Recognition, Optimization Algorithms.

1. Introduction

Detecting faces inside images by scanning the image for any faces is an algorithm called face detection.

In order to recognize faces, researchers have used various methods to extract features of human faces, it is important to extract the facial features that can adapt to environmental changes and improve the robustness of the recognition results.

Digital images are analyzed to find faces within them and then use image processing to establish the position of the face and identify its characteristics. Face recognition system includes face detection, face position, identity recognition, image preprocessing, etc. The distinction between face detection and recognition is that in detection, we just need to establish whether or not there is a face in the image, but in recognition, we want to know whose face it is.

Deep learning techniques have lately been directed to be used in computer vision applications by developers owing to the real progress it has made in attaining improved performance in face identification and achieving the best outcomes

in fixing challenges and difficulties with face detection [1].

The use of neural networks as a classification technique has become increasingly popular. Neural networks are a possible replacement for a variety of traditional classification techniques.

An activation function is a surprisingly simple mathematical equation that determines whether or not a neuron fires. This indicates that the activation function disables neurons whose inputs are irrelevant to the neural network's overall application.

When mapping inputs to outputs, an optimization algorithm determines the value of the weights that minimizes error. The accuracy of the deep learning model is greatly influenced by these optimization algorithms. They also have an impact on the model's training speed, total loss, and accuracy.

Our proposed method is to recognize faces using convolutional neural networks. Usually, initial layers of convolutional neural networks capture basic input image features like colors pattern, boundaries, and spots that are inattentive by the deeper hidden layers to form complex higher-level feature patterns to present a better-of-image illustration.

Each layer of the convolutional neural network output acts as an activation unit for the input images. Features are extracted from the fully-connected layers right before the final output classification layer. Considering this motivation, we extracted the features from the fully-connected layers of the network.

Inception-V3 is the factorization idea in the third iteration of GoogLeNet. The last fully-connected layer is used to extract the features from the Inception–V3 model to perform face classification.

VGG19 contains a stack of convolutional layers followed by three fully-connected layers, and features were extracted from the last three fully-connected layers.

The motivation for using pre-trained convolutional neural networks as feature extraction is that it doesn't demand a lot of computational capacity, and it is quite robust as we do not need to retrain the network, these attributes compel us to start with the feature extraction approach to classifying faces.

In this paper, we will study the impact of using different optimization algorithms with different activation functions depending on various embedding results of deep learning models.

The rest of this paper is arranged as follows: In section II, we discuss the related work, and in section III we present the implemented models of feature extraction. The used dataset is introduced in section IV, while section V compares the implemented algorithm's performance in detail. In section VI we discuss the obtained results and we conclude in Section VII with ideas about future vision.

2. Related Work

Researchers in the research paper [2] use a ResNet architecture, which introduces the Angular-Softmax loss to learn discriminative facial features with angular margins. The authors obtained 99.42% of accuracy on LFW and 95.0% on YTF by applying the nearest neighbor classifier with cosine similarity.

A new supervision signal, called center loss, is proposed for face recognition tasks in 2016. It learns a deep face feature center for each class while reducing the distance between the feature and the matching class center. Therefore, the discrimination of learned facial features is enhanced, and the variation of features in the intra-class is minimized [3]. They achieved an accuracy of 99.28% and 94.9% on LFW [4] and YouTube [5] Faces datasets, respectively.

Zhou et al. (2015) built the Megvii system, which is a deep learning-based face recognition system. They designed a simple 10 layers-deep convolutional neural network for recognition. Four face regions are cropped for representation extraction. Using the standard multi-class classification framework, they train their networks on the MFC database. During the testing phase, a PCA model is employed to reduce feature size, and a basic L2 norm is utilized to

measure the pair of testing faces. They achieved 99.50% accuracy on the LFW benchmark [6].

Lu and Tang proposed a principled multi-task learning approach based on Discriminative Gaussian Process Latent Variable Model, named GaussianFace, to enrich the diversity of training data. They achieved 98.52% accuracy on the Labelled Face benchmark (LFW) [7].

Ali and Kumar illustrate the performance of the inception model and squeeze net on the most famous classification algorithms, they obtained 94.87% of accuracy based on the logistic regression with ridge regularization when they depend on the results of the Inception model in the embedding step [8].

In this study, we provide a new method that uses the penultimate layer of the inception model for feature extraction with the neural network using SGD, Adam, and L-BFGS-B optimization algorithms with three types of activation functions, in the feature extraction process, we have used Inception V3, Squeeze Net and VGG19, and then we have applied neural network on the results of the feature extraction stage. We have made a comparison between all of them.

3. Feature Extraction

The content of images is a bunch of Meta information like image name, image size, image width, and image height in pixels, nothing helps with machine learning.

We need image descriptors, which are numbers that describe the content of these images. It is required to study the face in order to extract its features, in order to recognize this face, this technique is known as feature extraction because it generates numbers that describe faces and then stores the data in a database. Facial features go into different types:

The region type depends on observing people's mouths and eyes which are extremely important for grasping information and feelings. Therefore, in many applications, it may be very useful to automatically extract the eyes and mouth from the human face.

The key-point (Landmark) type provides a more accurate and consistent representation for alignment purposes, compared with region-based features. The contour feature extraction has higher complexity and computational burden than the key point, which matches texture and shapes at the same time.

Feature extraction is extensively used in the image processing industry, and it has various applications, particularly in the computer vision sector, since it uses algorithms to separate the elements or shapes in the picture.

3.1. Inception Model

There are four editions of the Inception Model, each new edition is a step forward from the preceding one. With the inclusion of batch normalization in the second edition, the architecture was enhanced to maximize accuracy while decreasing computing complexity, authors provide factorization concepts in the third edition, which are employed in this study. In the fourth edition, authors make the modules more uniform, because they noticed that some of the modules were more complicated than necessary.

Authors incorporated all of the upgrades of inception edition v2, to produce the third edition and in addition, they used RMSProp Optimizer with BatchNorm in the Auxiliary Classifiers.

Convolutions are factorized into smaller convolutions to reduce the number of parameters and factorization into asymmetric convolutions i.e. 1 layer of 3x3 filter will be 2 layers of 3x1 and 1x3 filter, parameters will be reduced from 9 to 6 parameters. They use factorization in the third version to reduce the number of parameters/connections while maintaining network performance, making the network less likely to overfit. The latest layer of the network is a fully connected layer followed by the Softmax layer to try to make a prediction. The model's performance is heavily influenced by factorization [9].

Inception Networks have proven to be more computationally efficient than VGGNet, both in terms of the number of parameters produced by the network and the consumption of memory and other resources.

We use vectors to represent images by applying the penultimate layer activation of the model. For each image, the model provides a vector of 2048 features, enabling us to compare and compute the similarity of the images.

3.2. Squeeze Net

It is comprised of "squeeze" and "expand" layers. It is an 18-layer deep model with 50x fewer parameters than AlexNet. It achieves nearly the same degree of accuracy. It has 1×1 filters only, therefore it operates as a fully-connected layer on feature points in the same place. One of its benefits, as the name implies, is that it reduces the depth of the feature map. When the depth is reduced, the following 3×3 filters in the expand layer have less computation to execute. It increases the speed since a 3×3 filter requires 9 times the calculation of a 1×1 filter.

It consists of a number of fire modules and a few pooling layers stacked on top of each other. The squeeze and expand layers maintain the same feature map size, but the former reduces the depth to a lower number while the latter increases it. In neural networks, squeezing (bottleneck layer) and expansion behavior is prevalent [10].

Each image is represented in a vector that contains 1,000 features, enabling us to compare and compute the similarity of the images.

3.3. VGG 19

VGG19 is a VGG model version that consists of 19 layers (16 convolution layers, 3 fully connected layers, 5 MaxPool layers, and 1 SoftMax layer). Instead of using huge filters, VGG uses smaller filters 3×3 with better depth. It has the same effective receptive field as if there were only one 7×7 convolutional layer. It is made up of two Fully Connected layers, each with 4096 channels, followed by another Fully Connected layer with 1,000 channels to predict 1000 labels [11].

4. Dataset

We have collected students' pictures of Syrian Private University and merged them with the Pins Face Recognition dataset which was collected from Pinterest. There are 115 identities and 18,036 faces.

We have used 70% of the data for learning and 30% for testing. Images cover considerable variation of expressions, age, and background clutter are supported by a huge number of images as illustrated in Fig. 1.

Fig. 1: Sample of Pins Face Recognition dataset.

5. Face Recognition

We have applied optimization algorithms with three types of activation functions, depending on the results of the embedding stage for the Inception model, VGG19, and Squeeze Net.

Node layers in neural networks include an input layer, one or more hidden layers, and an output layer. Each node, or artificial neuron, is linked to another and has a weight and threshold connected with it. If any individual node's output exceeds the defined threshold value, that node is activated and begins transferring data to the next layer of the network. Otherwise, no data is sent to the next network layer. Neural networks develop and improve their accuracy over time by using training data. However, once these learning algorithms have been fine-tuned for accuracy, they transform into powerful in-

struments in computer science and artificial intelligence, allowing us to rapidly categorize and cluster data.

We have used two types of optimizers which are algorithms used to change the attributes of the neural network in order to decrease the losses. We have used these optimizers with three activation functions for each one.

5.1. Stochastic Gradient Descent (SGD)

It is a first-order technique that makes an attempt to update the model's parameters more regularly. Following the computation of loss on each training example, the model parameters are changed.

We have applied SGD as optimizer three times, each one with a different activation function, depending on the embedding results of the inception model and squeeze net.

We have used the most famous activation functions:

Sigmoid Function

It is one of the most widely used activation functions and is common as an S-shaped curve which is also known as the Logistic function.

The values of the sigmoid function transform between the range 0 and 1 as shown in Fig. 2.

The mathematical expression for function is as follow:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{1}$$

The logistic function is computationally expensive due to the exponential nature of the function, and Using SGD optimizer with it is not suitable for a multilayer network. It has a Vanishing gradient which is clear in the function plot, which means that when inputs go smaller or larger, the function saturates at 0 or 1, with a derivative that's very near to 0, it obtains 3.7% and 3.5% of accuracy for 50 neurons with 2 layers network, after the inception model and squeezes net, respectively.

Hyperbolic Tangent

It is very similar to the logistic function, but it is symmetric around the origin, and the values range from. This function is defined as:

$$tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2}$$

It's continuous and differentiable at all points and the gradient is steeper. The next layers' inputs will not always be of the same sign since it is zero-centered.

When compared to the sigmoid function, the tanh function has a steeper gradient.

Table 1 illustrates the performance of a one-layer neural network with SGD optimizer and sigmoid activation function, depending on the number of neurons and the results of the embedding stage with different deep learning feature extraction methods (inception model, Squeeze Net, and VGG19).

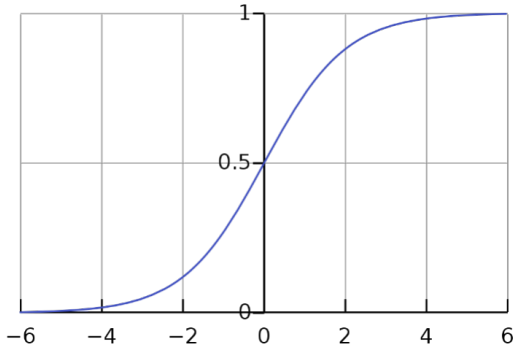

Fig. 2: The Plot of Logistic Function.

Table 2 illustrates the performance of the neural networks with SGD optimizer and Hyperbolic Tangent (tanh) activation function, depending on the number of neurons, the number of layers, and the results of the embedding stage with different deep learning feature extraction methods (inception model, Squeeze Net, and VGG19).

Tanh is recommended over the sigmoid function because it is zero-centered and gradients are not constrained to flow in a specific direction.

We realize that applying a neural network with 4 layers and 50 neurons achieves 77.5% of

Tab. 1: Neural network performance with the SGD optimizer and sigmoid activation functions.

Neurons	Inception Model				Squeeze Net				VGG 19			
	Accuracy	F-measure	Precision	Recall	Accuracy	F-measure	Precision	Recall	Accuracy	F-measure	Precision	Recall
10	6.9	2.2	4.4	6.9	6.8	2.3	3.7	6.8	6.2	2.6	5	6.2
20	11.3	6.2	11.7	11.3	11.1	6.6	10	11.1	10.4	6.2	8	10.4
30	14.8	10.1	16.6	14.8	15.9	11.1	15.7	15.9	15.4	10.5	14.7	15.4
40	20.4	16	22.4	20.4	19	14.3	21	19	18.4	13.7	18.4	18.4
50	24.3	19.6	27.6	24.3	21.7	17.1	23.2	21.7	20.9	16.4	22.7	20.9

Tab. 2: Performance of Neural Network with SGD Optimizer and tanh activation function for different neurons and layers count.

Layers	Neurons	Inception Model					Squeeze Net					VGG 19				
		Accuracy	F-Measure	Precision	Recall	Elapsed Time	Accuracy	F-Measure	Precision	Recall	Elapsed Time	Accuracy	F-Measure	Precision	Recall	Elapsed Time
1	20	43.1	40.4	43.1	43.1	178	31	27.6	31	31	171	38.1	35	38.5	38.1	183
	30	56.7	55.3	56.7	56.7	196	40.7	38.7	40.9	40.7	182	50	48.2	50.3	50	201
	40	66.1	65.3	66.3	66.1	210	46.1	44.6	45.8	46.1	200	58	57	58.4	58	215
	50	73.3	72.9	73.3	73.3	223	50.8	49.7	50.4	50.8	214	64.3	63.6	64.4	64.3	228
2	20	34.8	30.4	33.9	34.8	217	25.2	21	22.2	25.5	210	34.7	30.1	35.1	34.7	222
	30	51.7	49.4	51.6	51.7	234	34.7	31.1	34.3	34.7	216	48.5	45.8	49	48.5	240
	40	63.2	61.9	63.3	63.2	245	41.8	39.3	41.2	41.8	222	57.4	55.9	57.6	57.4	251
	50	69.9	69.3	69.9	69.9	258	46.5	44.5	45.9	46.5	233	65.6	64.8	66	65.6	266
3	20	31.4	25.9	32.5	31.4	228	23.5	18.9	20.1	23.5	212	33.1	28	32.5	33.1	233
	30	50.5	47.9	51.2	50.5	249	31.5	28.1	31	31.5	224	46.3	43.4	46.7	46.3	250
	40	63.3	61.7	63.6	63.3	265	39.5	36.9	38.5	39.5	235	60.5	58.8	61.3	60.5	271
	50	74.8	74.2	75.2	74.8	273	44.7	42.3	43.6	44.7	248	68.6	67.8	68.9	68.6	280
4	20	25.5	19.2	25.9	25.5	241	20.1	15.1	16.5	20.1	222	28.8	23.6	27.5	28.8	247
	30	44.4	40	47	44.4	265	30.1	26.2	28.2	30.1	232	46	43	46.3	46	270
	40	64.5	62.6	64.7	64.5	290	35.5	32.1	35	35.5	248	59	57.2	59.7	59	294
	50	77.5	76.8	77.7	77.5	311	43.8	41.3	43.1	43.8	262	71.6	70.9	71.8	71.6	316

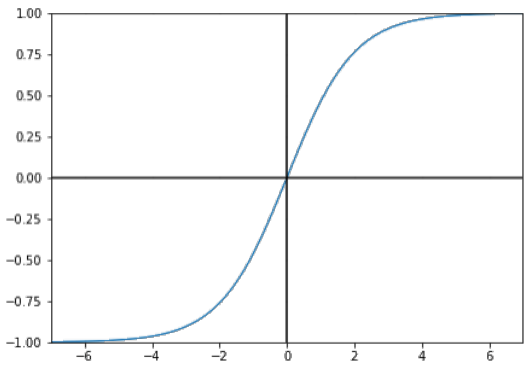

Fig. 3: The Plot of Tanh Function.

accuracy when the used activation function is Hyperbolic Tangent and the optimizer is SGD depending on the results of the inception model, it's a good result, but when we applied it with five layers the accuracy has been raised by 0.6%, which is not a big difference.

The accuracy decreased when we depended on the embeddings of VGG 19 which achieved 71.6% of accuracy, and it takes 15 minutes, while depending on the results of Squeeze Net led to a significant decrease in accuracy, as it achieved 43.8% of accuracy, although it was faster than both.

Rectified Linear Unit (ReLU)

It returns the value back for any positive input and returns 0 only if the input is negative, its derivative is 0 for any negative input, so it is not a differentiable function, but it is a continuous function.

The function is defined as:

$$f(x) = max(0, x), \begin{cases} 0, & x \leq 0 \\ 1, & x > 0 \end{cases} \tag{3}$$

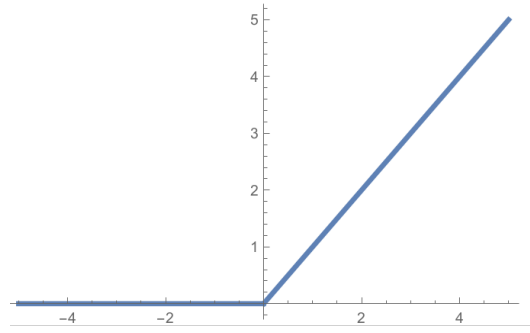

Fig. 4: The Plot of ReLU Function.

Because the derivative of the ReLU function is 0 for any negative input, it is not differentiable. Gradient Descent is helped by the fact

that ReLU's output has no maximum value (it is not saturated).

The last implementation of SGD optimizer with ReLU activation function, depending on the embedding results of the inception model, achieved the best results, as shown in Tab. 3.

In this scenario, we have achieved 93.6% of accuracy with 40 neurons and 4 layers, ReLU function is very fast to compute (compared to sigmoid and tanh) and it works very well with deep neural networks, especially when we use it depending on the embedding results of inception model, which gives 2,048 features.

When we depended on the squeeze net's embedding results, the accuracy extremely decreased since it has 1,000 features only (compared to the inception model).

We realize that using SGD optimizer is suitable for multi-layer neural networks, where the accuracy increased when the number of layers increased depending on the inception model or VGG 19 model, while the accuracy didn't affect when the embedding model is squeeze net.

5.2. Adaptive Moment Estimation (Adam)

It is a gradient-based optimization approach for stochastic objective functions. It combines the benefits of two SGD extensions: Root Mean Square Propagation (RMSProp) and the Adaptive Gradient Algorithm (AdaGrad).

It is a first-order technique that computes adaptive learning rates for each parameter and can be used instead of the classical stochastic gradient descent procedure to update network weights iterative based on training data.

The main difference between Adam and Stochastic Gradient Descent is that SGD does not change the learning rate (alpha) during training and maintains a single learning rate for all weight updates.

The following equations illustrate the first and second order of momentum:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{4}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{5}$$

The gradient and squared gradient are calculated using an exponential moving average, and the decay rates of these moving averages are controlled by the parameters beta1 and beta2.

Moment estimations are biased towards zero when the moving averages' initial value and beta1 and beta2 values are near to 1.0. This prejudice is eliminated by computing biased estimates first, then bias-corrected estimates.

We have applied Adam as optimizer three times, each one with a different activation function, depending on the results of the embeddings stage.

Firstly, we have used the sigmoid function, depending on neurons count, layers count, and the results of the embedding model. Table 4 illustrates the performance of a neural network with Adam optimizer and sigmoid function.

We notice that the results of one layer are better than the results of 2 or 3 layers since it achieved 78.6% of accuracy by 20 neurons only and depending on the inception model. Secondly, increasing the number of layers leads to a significant decrease in accuracy whatever the embedding model. Table 5 illustrates the performance of a neural network with Adam optimizer using tanh activation function.

We realize that using the Adam optimizer with sigmoid or tanh function is not suitable for multi-layer neural networks, even if the embedding step depends on the inception model, VGG19, or squeeze net. However, using the inception model is better than squeeze net and VGG19 as shown in the previous table.

Finally, we have achieved 89.1% of accuracy by using one layer neural network with 20 neurons, ReLU function, and Adam optimizer, the accuracy decreased to 83.6% when we use 3 layers network as shown in Tab. 6.

We realize that using the embedding results of the inception model is better than the embedding results of the squeeze net and VGG19, since we obtained better accuracy using the inception model.

Tab. 3: Performance of Neural Network with SGD Optimizer and ReLU activation function for different neurons and layers count.

Layers	Neurons	Inception Model					Squeeze Net					VGG 19				
		Accuracy	F-Measure	Precision	Recall	Elapsed Time	Accuracy	F-Measure	Precision	Recall	Elapsed Time	Accuracy	F-Measure	Precision	Recall	Elapsed Time
1	10	32.9	30.3	30.4	32.9	154	23.6	20.5	20.3	23.6	135	26.8	23.5	23.8	26.8	161
	20	52.5	51.4	51.2	52.5	168	38.2	36.6	36.4	38.2	148	44.3	43	43.1	44.3	175
	30	67.5	67.1	67.1	67.5	187	45.7	44.6	44.4	45.7	172	56.2	55.4	55.5	56.2	194
	40	76.9	76.6	76.7	76.9	202	51.2	50.5	50.3	51.2	187	63.8	63.4	63.5	63.8	210
2	10	31.9	29	29.6	31.9	188	22.9	19.5	19.2	22.9	168	28.5	24.8	25.4	28.5	194
	20	56.8	55.3	55.3	56.8	207	37.4	35.8	35.8	37.4	184	48.7	47.3	47.5	48.7	215
	30	71.5	71.1	71.3	71.5	225	45.4	44.3	44.5	45.4	207	61.4	60.8	61	61.4	231
	40	84.6	84.4	84.6	84.6	238	51.8	50.9	50.9	51.8	231	69	68.6	68.9	69	244
3	10	30.4	27	27.4	30.4	197	20.1	16.2	18.2	20.1	185	28.1	24.4	25.1	28.1	205
	20	56.4	55.3	56	56.4	220	35	33.1	33.3	35	209	49.4	48.2	48.9	49.4	228
	30	78.7	78.4	78.7	78.7	237	43.6	42.2	42.9	43.6	229	66.2	65.8	66.7	66.2	245
	40	91	91	91.3	91	255	51	49.8	50	51	247	78.2	78	78.4	78.2	261
4	10	29.5	25.5	28	29.5	220	20.7	16.9	17.9	20.7	206	24.5	21.2	21.5	24.5	230
	20	59.9	59	60.4	59.9	231	32.8	30.4	30.7	32.8	230	45.7	43.7	45.3	45.7	242
	30	82.2	82	82.7	82.2	257	43.1	41.3	42.1	43.1	244	67.1	66.5	67.5	67.1	269
	40	93.6	93.7	94	93.6	275	50.7	49.5	50.4	50.7	267	77.7	77.4	78.3	77.7	290

Tab. 4: Performance of Neural Network with Adam Optimizer and sigmoid activation function for different neurons and layers count.

Layers	Neurons	Inception Model					Squeeze Net					VGG 19				
		Accuracy	F-Measure	Precision	Recall	Elapsed Time	Accuracy	F-Measure	Precision	Recall	Elapsed Time	Accuracy	F-Measure	Precision	Recall	Elapsed Time
1	5	17.1	12	16.9	17.1	223	14.2	9	9.9	14.2	204	15.8	10.9	12.6	15.8	235
	10	41.3	39.1	41.5	41.3	259	28.8	25.5	25.7	28.8	242	36.4	33.7	34.4	36.4	271
	15	61.1	60.3	61	61.1	282	39.1	37.1	37	39.1	259	54.2	53	53.2	54.2	296
	20	78.6	78.3	78.5	78.6	305	46.3	44.8	44.6	46.3	286	67.4	66.9	67.1	67.4	319
2	5	8	4.1	5.4	8	266	9	4.5	6	9	242	9.3	4.5	4.4	9.3	279
	10	21.1	16.4	19.6	21.1	298	19.2	14.9	16.2	19.2	272	21.8	17.4	20.6	21.8	311
	15	37.6	34.1	38.1	37.6	314	27.3	23.5	23.3	27.3	285	36.1	33.2	35.5	36.1	327
	20	56.9	55.6	57.4	56.9	330	34.3	31.5	31.7	34.3	301	49.3	47.1	48	49.3	344
3	5	4.8	1	0.7	4.8	319	5.3	1.7	1.4	5.3	295	5	1.3	0.9	5	332
	10	12.7	6.9	5.9	12.7	344	8.5	4.7	4.6	8.5	324	9	3.9	3.6	9	358
	15	21.3	15.4	18.7	21.3	375	12.9	8.5	8.4	12.9	347	14.3	8.9	8.6	14.3	372
	20	26.7	20.7	22.4	26.7	392	19.3	15.1	16	19.3	371	24.3	18.4	18.6	24.3	401

Tab. 5: Performance of Neural Network with Adam Optimizer and tanh activation function for different neurons and layers count.

Layers	Neurons	Inception Model					Squeeze Net					VGG 19				
		Accuracy	F-Measure	Precision	Recall	Elapsed Time	Accuracy	F-Measure	Precision	Recall	Elapsed Time	Accuracy	F-Measure	Precision	Recall	Elapsed Time
1	5	18.1	13.7	20.2	18.1	191	16	12.2	11.8	16	172	16.6	12.6	15.1	16.6	203
	10	44.9	43.3	45.3	44.9	206	31.2	28.3	29.1	31.2	188	41.3	39.2	40.5	41.3	218
	15	67.2	66.8	67.1	67.2	221	41.4	40.2	40.1	41.4	201	59	58.5	58.6	59	238
	20	85	85	85	85	242	49.5	48.5	48.4	49.5	220	73.6	73.3	73.4	73.6	259
2	5	12.1	6.7	9.4	12.1	205	13.2	8.5	9.8	13.2	183	10.5	5.8	7.2	10.5	218
	10	32.6	30.1	34.6	32.6	212	26.8	24.2	24.3	26.8	204	30.3	26.8	29.9	30.3	229
	15	58.2	57.5	58.6	58.2	233	36.8	35	35	36.8	221	49.8	48.8	49.1	49.8	244
	20	74.7	74.5	74.9	74.7	246	44.8	43.4	43.1	44.8	235	66.1	65.5	65.7	66.1	261
3	5	8.5	4	3.6	8.5	215	11.3	6.7	7.6	11.3	199	9.4	4.3	5.5	9.4	230
	10	30.2	26.1	31	30.2	238	23.7	20.1	20	23.7	218	24.9	21.4	25	24.9	256
	15	47	45.6	49.2	47	249	34.3	31.8	32.1	34.3	236	41.2	39.5	40.7	41.2	271
	20	66.9	66.5	67.5	66.9	271	42	40.4	40.3	42	251	56.5	55.5	56	56.5	287

Tab. 6: Performance of Neural Network with Adam Optimizer and ReLU activation function for different neurons and layers count.

Layers	Neurons	Inception Model					Squeeze Net					VGG 19				
		Accuracy	F-Measure	Precision	Recall	Elapsed Time	Accuracy	F-Measure	Precision	Recall	Elapsed Time	Accuracy	F-Measure	Precision	Recall	Elapsed Time
1	5	24.1	21	21.7	21	177	18.9	15.7	15.7	18.9	175	23.2	19.8	19.8	23.2	190
	10	49.5	47.9	48.2	49.5	191	33.8	31.7	31.6	33.8	189	44.7	43	43.2	44.7	203
	15	70.2	69.6	69.8	70.2	207	44	42.7	42.6	44	209	62.2	61.7	61.9	62.2	230
	20	89.1	88.9	89	89.1	227	51.3	50.1	50.1	51.3	226	76.2	75.8	76.2	76.2	242
2	5	21.6	17.7	19.1	21.6	189	17.3	14.2	13.8	17.3	180	20.1	16.5	16.9	20.1	201
	10	46.3	44.4	44.8	46.3	197	33.3	31.2	31.4	33.3	200	40.4	38.5	39.2	40.4	218
	15	68.3	67.6	68.2	68.3	210	42.1	40.6	41.2	42.1	216	59.2	58.2	58.8	59.2	226
	20	87.7	87.5	87.8	87.7	232	49.7	48.6	49.3	49.7	232	74.6	74.3	74.9	74.6	249
3	5	19.1	14.4	14.9	19.1	203	17.5	14.6	15.2	17.5	191	17.8	14.2	15.9	17.8	217
	10	44.4	42.6	44.4	44.4	218	29.7	27.3	28	29.7	204	39.8	37.3	37.6	39.8	234
	15	66.9	66.1	66.8	66.9	231	40.1	38.5	38.9	40.1	224	54.8	53.7	54.5	54.8	250
	20	83.6	83.5	84	83.6	247	47.8	46.7	47.4	47.8	239	71.3	70.9	71.5	71.3	276

The accuracy decreased slightly when the number of layers increased in the network, no matter the embedding model or the neurons count.

5.3. L-BFGS-B

t is the most widely used second-order technique for numerical optimization, for multivariate objective functions, it employs the second-order derivative, also known as the Hessian matrix. It is cost-effective for very large issues, and it is part of a group of algorithms known as Quasi-Newton methods.

Table 7 illustrates the performance of a neural network with L-BFGS-B optimizer and sigmoid function.

Using sigmoid function with L-BFGS-B optimizer is not suitable for multi-layer neural network. The accuracy decreased when the number of layers increased whatever the number of neurons or the embedding model.

L-BFGS-B optimizer with sigmoid function obtained 75% of accuracy with one layer network depending on the results of inception model, but it has fallen dramatically to 54.1% when the neural network had 2 layers.

Table 8 illustrates the performance of a neural network with L-BFGS-B optimizer and Hyperbolic Tangent (tanh) activation function, depending on the results of the embedding step, the number of neurons, and the number of layers.

One layer neural network with 40 neurons, achieved 88.5% of accuracy using tanh function with L-BFGS-B optimizer.

Using ReLU function with L-BFGS-B optimizer achieved 94% of accuracy with 20 neurons and one layer as shown in Tab. 9.

6. Discussion of Results

Depending on the results aforementioned, we noticed that the obtained results depending on the embedding results of the inception model is better than using VGG19 or squeeze net as shown in Fig. 5.

We have obtained best accuracy when using ReLU function with any of optimizers and depending on the inception model.

Using SGD algorithm with tanh function or ReLU function is suitable for multi-layer neural network, where the accuracy increased when the number of layers increased depending on the inception model or VGG 19 model.

The achieved accuracy by using SGD optimizer with ReLU function increased well by increasing layers count when we depend on embeddings of inception model, but the accuracy didn't affected by the increase of layers count with squeeze net as shown in Fig. 6.

Whatever activation function is used with Adam's optimizer, there is no significant effect of increasing the number of layers. Figure 7 illustrate the impact of layers counts on a neural network with Adam optimizer and ReLU activation function.

When compared to the second order methods such as L-BFGS, the first order methods are faster and easier to implement, but this comes with a cost.

Second order techniques get you closer to the optimal than first order methods, but the cost every iteration is enormous. Furthermore, parameter tuning has always been a difficulty with first order techniques.

Using L-BFGS-B optimizer is suitable with one layer neural network, the accuracy decreased with multilayer neural network whatever the activation function. Figure 8 shows the impact of increasing layers count.

7. Conclusions

We have made a combination of deep learning models for feature extraction with Neural Networks using the most famous optimization algorithms with three types of activation functions.

We have tried to focus on the performance changes for each implementation, using Neural

Tab. 7: Performance of Neural Network with L-BFGS-B Optimizer and sigmoid activation function for different neurons and layers count.

Layers	Neurons	Inception Model					Squeeze Net					VGG 19				
		Accuracy	F-Measure	Precision	Recall	Elapsed Time	Accuracy	F-Measure	Precision	Recall	Elapsed Time	Accuracy	F-Measure	Precision	Recall	Elapsed Time
1	10	28.6	26.1	28.8	28.6	204	22.6	19.4	19.6	22.6	198	19.9	16.6	18.6	19.9	218
	20	47.3	46.4	47.1	47.3	211	38.6	36.8	36.5	38.6	205	35.7	34.4	34.4	35.7	225
	30	60.3	59.9	60.3	60.3	219	53.4	52.5	52.2	53.4	213	53	52.3	52.2	53	237
	40	75	74.9	74.9	75	228	62.6	62	61.8	62.6	222	63.7	63.3	63.2	63.7	245
2	10	12.9	8.7	12.6	12.9	209	17.4	13.3	14.2	17.4	203	12.7	9	10.8	12.7	229
	20	31.5	29.6	31	31.5	220	28.6	26.7	27.3	28.6	211	25.6	23.6	24.5	25.6	238
	30	42.6	41.6	42.8	42.6	231	35.3	33.7	33.7	35.3	220	35.6	34.4	34.7	35.6	250
	40	54.1	53.5	53.7	54.1	241	45.7	44.6	44.4	45.7	229	42.9	42	42	42.9	266

Tab. 8: Performance of Neural Network with L-BFGS-B optimizer and tanh activation function for different neurons and layers count.

Layers	Neurons	Inception Model					Squeeze Net					VGG 19				
		Accuracy	F-Measure	Precision	Recall	Elapsed Time	Accuracy	F-Measure	Precision	Recall	Elapsed Time	Accuracy	F-Measure	Precision	Recall	Elapsed Time
1	10	26.6	24	27.2	26.6	167	22.2	19	19.9	22.2	154	23.2	20.1	22	23.2	193
	20	51.3	50.7	50.9	51.3	181	38.8	37.3	37	38.8	168	43.1	42	41.8	43.1	208
	30	73.8	73.7	73.7	73.8	195	52.9	52.1	51.9	52.9	181	62.9	62.5	62.4	62.9	224
	40	88.5	88.4	88.5	88.5	214	64.7	64.2	64	64.7	199	79.7	79.6	79.6	79.7	241
2	10	16.2	12.6	18.4	16.2	179	18.1	14.6	16.3	18.1	165	16	12.8	15.3	16	205
	20	39.4	38.5	38.9	39.4	185	32.7	31	31	32.7	171	31.9	30.3	30.7	31.9	213
	30	61.4	61.2	61.2	61.4	197	45.1	44	43.8	45.1	191	45.9	45.2	45.1	45.9	225
	40	80.1	80.1	80.1	80.1	217	58	57.3	57.2	58	206	60.8	60.4	60.3	60.8	246
3	10	16.6	12.8	17.9	16.6	189	17.7	14.4	15.2	17.7	175	12.9	9.7	12	12.9	219
	20	35.4	33.9	35.3	35.4	204	29.1	27.1	27.4	29.1	184	29	27.1	28.1	29	237
	30	57.9	57.5	57.8	57.9	218	42.5	41.5	41.4	42.5	196	44.8	44.1	44.1	44.8	253
	40	71.4	71.3	71.3	71.4	231	57.7	57	56.8	57.7	215	62.8	62.5	62.5	62.8	266

Tab. 9: Performance of Neural Network with L-BFGS-B optimizer and ReLU activation function for different neurons and layers count.

Layers	Neurons	Inception Model					Squeeze Net					VGG 19				
		Accuracy	F-Measure	Precision	Recall	Elapsed Time	Accuracy	F-Measure	Precision	Recall	Elapsed Time	Accuracy	F-Measure	Precision	Recall	Elapsed Time
1	5	22.7	19.3	20.1	22.7	172	19.7	16.6	16.4	19.7	158	21.1	18.2	18.9	21.1	201
	10	51	49.9	49.8	51	185	34.8	33	32.8	34.8	174	41.7	40	39.8	41.7	213
	15	73.6	73.1	72.9	73.6	202	44.5	43.2	42.9	44.5	188	59.6	58.6	58.2	59.6	230
	20	94	93.9	93.9	94	224	52.6	51.6	51.3	52.6	205	74.3	74	74	74.3	254
2	5	13.8	10.1	11.4	13.8	185	11.4	7.7	8	11.4	174	15.9	12.4	12.4	15.9	218
	10	39.7	37.8	37.9	39.7	195	30	28	27.8	30	185	37.5	35.3	35.2	37.5	224
	15	59.4	58.5	58.3	59.4	205	40.2	38.6	38.5	40.2	198	52.5	51.2	50.9	52.5	236
	20	82.5	82.2	82.1	82.5	228	48.3	47.1	46.8	48.3	213	66.4	65.6	65.4	66.4	260
3	5	8.1	3.9	3.7	8.1	199	9.6	6.4	6.7	9.6	186	5.4	1.9	1.5	5.4	228
	10	31.2	28.4	28.7	31.2	214	21.7	18.4	18.6	21.7	195	29.8	27	27.5	29.8	246
	15	45.2	43.1	42.9	45.2	228	32.9	30.7	30.5	32.9	208	40.3	38.4	38.6	40.3	259
	20	52.6	51.3	51	52.6	241	40.4	38.8	38.3	40.4	228	53	51.8	51.6	53	272

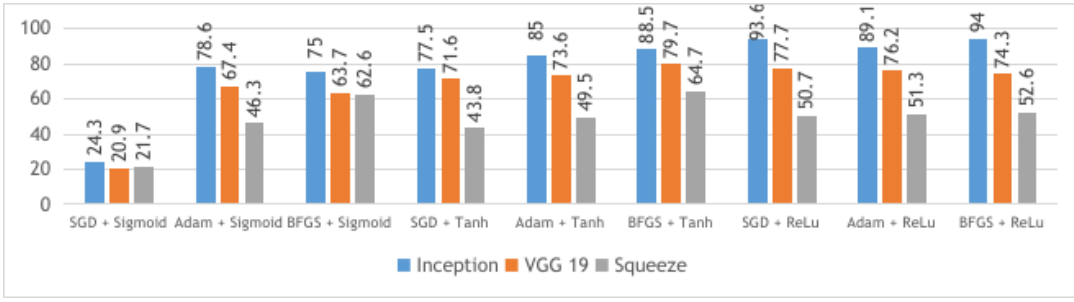

Fig. 5: Performance of optimizers and activation functions depending on embedding models.

Fig. 6: The impact of the feature extraction models with the layers counts on a Neural Network with SGD Optimizer and ReLU activation function.

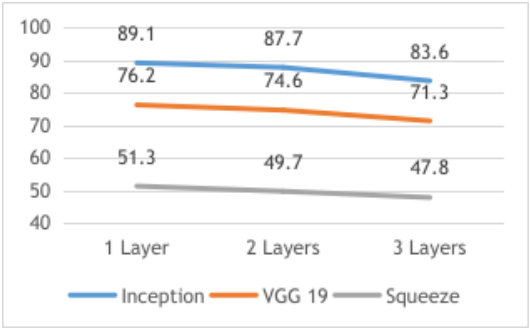

Fig. 7: The impact of the feature extraction models with the layers counts on a Neural Network with Adam Optimizer and ReLU activation function.

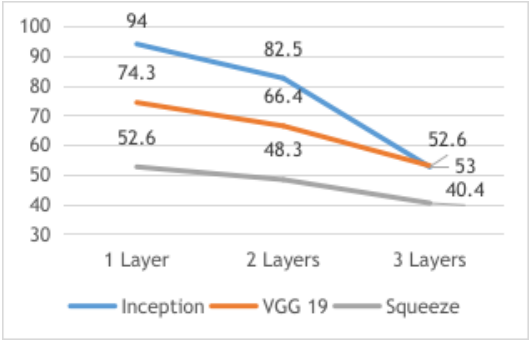

Fig. 8: The impact of the feature extraction models with the layers counts on a Neural Network with L-BFGS-B Optimizer and ReLU activation function.

Networks depending on the results of inception model for feature extraction obtained better accuracy than other feature extraction models.

We have noticed that SGD optimizer is suitable for multilayer neural networks, whereas using Adam optimizer didn't have a significant impact on multilayer neural networks. Finally, using L-BFGS-B optimizer is not suitable for multilayer neural networks where accuracy has fallen dramatically. In the future, we will combine convolutional neural networks (CNN) with support vector machines (SVM) by replacing the Softmax layer with SVM to obtain better results.

References

[1] Krizhevsky, A., Sutskever, I., & Hinton, G.E. (2017). Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, *60*(6), 84–90.

[2] Liu, W., Wen, Y., Yu, Z., Li, M., Raj, B., & Song, L. (2017). Sphereface: Deep hypersphere embedding for face recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 212–220.

[3] Wen, Y., Zhang, K., Li, Z., & Qiao, Y. (2016). A discriminative feature learning approach for deep face recognition. In *European conference on computer vision*, Springer, 499–515.

[4] Huang, G.B., Mattar, M., Berg, T., & Learned-Miller, E. (2008). Labeled faces in the wild: A database forstudying face recognition in unconstrained environments. In *Workshop on faces in'Real-Life'Images: detection, alignment, and recognition*.

[5] Wolf, L., Hassner, T., & Maoz, I. (2011). Face recognition in unconstrained videos with matched background similarity. In *CVPR 2011*, IEEE, 529–534.

[6] Zhou, E., Cao, Z., & Yin, Q. (2015). Naive-deep face recognition: Touching the limit of LFW benchmark or not? *arXiv preprint arXiv:150104690*.

[7] Lu, C. & Tang, X. (2015). Surpassing Human-Level Face Verification Performance on LFW with GaussianFace. In Bonet, B. & Koenig, S. (editors), *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, AAAI Press, 3811–3819.

[8] Ali, M. & Kumar, D. (2021). A Combination between Deep learning for feature extraction and Machine Learning for Recognition. In *2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, IEEE, 01–05.

[9] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2818–2826.

[10] Iandola, F.N., Han, S., Moskewicz, M.W., Ashraf, K., Dally, W.J., & Keutzer, K. (2016). SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size. *arXiv preprint arXiv:160207360*.

[11] Simonyan, K. & Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. In Bengio, Y. & LeCun, Y. (editors), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

About Authors

Mahmoud ALI hold his MSc degree from Syrian Virtual University in 2018. He is now a Ph.D. Scholar in Computer Engineering department at Marwadi University, Rajkot, India. His research interests are Semantic Web, Linked Open Data, Computer Vision, Machine Learning, and Deep Learning.

Dinesh KUMAR received his Ph.D. degree at the University of Coimbra, Portugal in 2015. He is now an Associate Professor in the Computer Engineering department at Marwadi University, Rajkot, India. His research interests are Signal/Image processing for biomedical applications, Pattern Recognition, and Machine Learning.